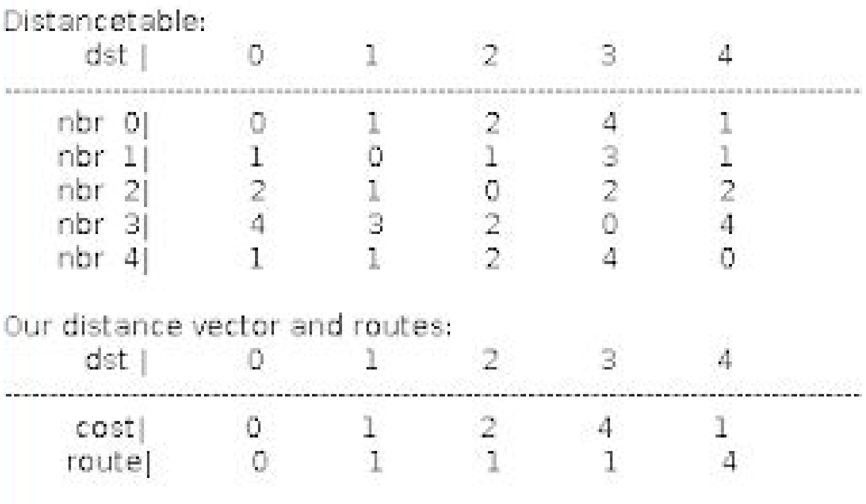
TDTS06

Lab 4 report

Mayeul Gasc (mayga330@student.liu.se)
Pierre Marsaa (piema262@student.liu.se)

For this lab, we need to implement the distance-vector algorithm for a router class that will allow each router to know its **distance to another** and the **route associated**. For this implementation we relied on the algorithm as it is described in the *4th edition of the course book*.



```
Distancetable:
     dst |      0      1      2      3      4
-----------------------------------------------
    nbr  0|     0      1      2      4      1
    nbr  1|     1      0      1      3      1
    nbr  2|     2      1      0      2      2
    nbr  3|     4      3      2      0      4
    nbr  4|     1      1      2      4      0

Our distance vector and routes:
     dst |      0      1      2      3      4
-----------------------------------------------
     cost|      0      1      2      4      1
    route|      0      1      1      1      4
```

Output example for a router

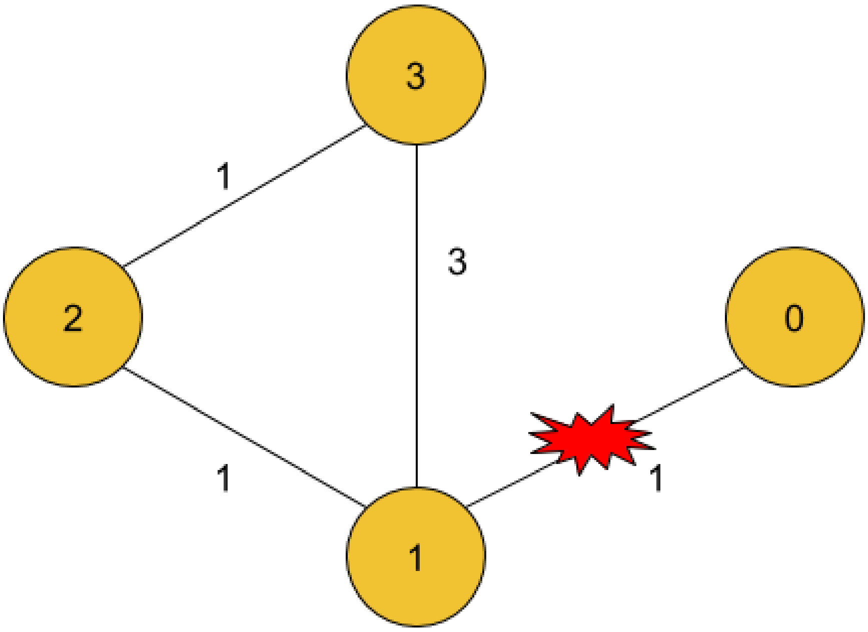
For each router, the program creates a GUI in which it gives the information in this position (example above). In particular, he gives his distance table, the route for each node in the network and the cost of each route.

Questions

(i) The distance-vector algorithm is an algorithm which, when deployed on a network of routers, allows them to know the route to any other router while knowing only the value of the adjacent links. For this, the adjacent nodes exchange their distance with all the other routers and update their distance table accordingly. Each time an update takes place, the router informs all its neighbors so that they also update themselves. These exchanges continue until the system stabilizes.

(ii) To test our algorithm, we simply used the examples given on the lab's web page, comparing our results to the optimal path that we can read.

(iii) Poisoned reverse algorithm is efficient but limited, in particular, it cannot manage multiple nodes loops (e.g. loop passing by N1 → N2 → N3 → N1 → etc.).



For example, in the drawing above if the link 0-1 fails, router 1 will refer to router 2 (or 3, but suppose 2 for the example), which thanks to poisoned reverse will not refer to 1 but to 3, but 3 will refer to 1 and we have a loop.

(iv) By looking on the internet, there are different solutions to this problem, for example the Routing Information Protocol which avoids routing loops by fixing a limit to the number of hops possible for an update.

References

- Kurose, J. and Ross, K., 2017. *Computer Networking*. 7th ed. Boston (Mass.): Pearson, pp.416-420.
- T. Cegrell, "A Routing Procedure for the TIDAS Message-Switching Network," in *IEEE Transactions on Communications*, vol. 23, no. 6, pp. 575-585, June 1975, doi: 10.1109/TCOM.1975.1092864.
- RFC 1058